SURFACE COMPRESSION BASED ON REFERENCE GEOMETRY IN ANIMATION MODELS

Inventor: Scott B. Peterson

Field of the Invention

[0001] The present invention relates to the compression and storage of animation models, and more particularly, to the compression of the surface geometry of animation models to reduce storage requirements in animation sequences involving large numbers of models.

Background of the Invention

[0002] Three-dimensional character animation requires significant storage requirements. To animate a single 3D character (or generally any model) over some interval of time typically requires storing the position of the character at the time of each frame, in order to be able to render the frame. Complex animation sequences can involve hundreds of characters. For example, an animation sequence of á crowd containing 200 different animation cycles would require the storage of the complete position information for each of the 200 characters, for each frame of their respective animation cycles. A typical character cycle may contain thousands of points and hundreds of frames of animation, and the storage requirements for a lengthy cycle can easily reach hundreds of megabytes of storage. Over the course of a feature length animation, these storage requirements quickly mushroom and can easily reach terabytes of storage.

[0003] Accordingly, it is desirable to provide a way of compressing character information in animation sequences to reduce storage requirements for character

models, and thus by extension, reduces the storage needed for crowds or other complex animation sequences.

Summary of the Invention

[0004] The present invention provides a methodology and system for compressing animation models. The invention is particularly well suited for compressing the information describing the poses of a model during an animation sequence, which may last any number of frames.

[0005] Generally, the methodology uses an initial or reference model for selected frame of an animation sequence, and based on the reference model, predicts the an offset model in each of a number of subsequent frames. The prediction for the offset model at a subsequent frame is based on the relative position of control vertices on the surface of the model of the reference frame, as those relative positions are mapped onto the surface of the offset model at the later, frame. The differences between the predicted vertices for the offset model and the actual vertices of the offset model are determined. These differences are preferably compressed for storage. The difference information takes a small number of bytes relative to the amount of data to store the actual control vertices information. This prediction process is repeated for each of the subsequent frames (e.g., 100 to 200 frames), using the initial reference frame as the prediction reference. Thus storage savings are achieved directly since the entire model data needs to be stored only for the reference pose, as the subsequent poses are stored in terms of the differences between the reference pose and the predictions for the poses of the offset models. The compression of the difference information yields additional storage savings.

[0006] From the compressed model information, any frame of the animation sequence can be retrieved and decompressed, using just the reference

frame information and the compressed model for the desired frame. Generally, the control vertices of the reference surface are used to predict the locations of the corresponding vertices on the desired offset model. The compressed prediction differences are decompressed and added to the predicted locations to obtain desired final offset control vertices, thereby reconstructing the model. The degree of error between the original offset model and the reconstructed model is controlled based on quantization applied to the coordinates of the reference control vertices and the differenced between the predicted and actual offset locations during compression.

[0007] The present invention has embodiments in various software products including geometry compressor, and a geometry decompressor, as well as in the compressed geometry files themselves. Also, the present invention may be embodied in various systems that include computers executing the geometry compressor and/or geometry decompressor, and preferably data compression and decompression algorithms.

Brief Description of the Drawings

[0008] FIG. 1 is an illustration of a typical NURBS model.

[0009] FIG. 2 is a detail of the surface of a model at a given frame.

[0010] FIG. 3 is a detail of the same surface of the model following a deformation.

[0011] FIG. 4 is system diagram for an embodiment of the invention.

[0012] FIG. 5 is a data flow diagram showing the data flow in one embodiment of the compression process.

[0013] FIG. 6 illustrates a traversal pattern for traversing control vertices in one embodiment.

[0014] FIG. 7 illustrates seed vertices in one embodiment.

[0015] FIG. 8 illustrates how basis triangles are selected in one embodiment.

[0016] FIG. 9 illustrates the prediction of an offset vertex in one embodiment.

[0017] FIG. 10 is a data flow diagram showing the data flow in one embodiment of the decompression process.

[0018] The figures depict a preferred embodiment of the present invention for purposes of illustration only. One skilled in the art will readily recognize from the following discussion that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the invention described herein.

Detailed Description of the Invention

[0019] Referring to Figs. 1-3 there is shown various illustrations of a model, as a basis for explaining the operation of the invention. Fig. 1 illustrates an example model 100, here a character's head, based on a NURBS type surface geometry. The model is a plurality of surfaces 102, each surface is composed of a matrix of control vertices 104. Figs 2 and 3 illustrate in detail a portion of the model 100, showing a portion of the surface 102 and several vertices 104. Briefly, a surface 102 is described by matrix M of $i \times k$ control vertices P, where each vertex P defines a coordinate (x,y,z) in the local space of the model.

[0020] Fig. 2 illustrates a pose (or position) of the model at some reference time, and hence is referred to as the reference pose or reference model; by extension each of the control vertices 104 has a particular position at this reference time. The reference time is determined by the animator, and may be understood by analogy to correspond to a snapshot of the model at one particular time during the animation, typically the first frame.

[0021] Fig. 3 illustrates the position of the same portion of the model 102 at a later time after the reference time; this position is referred to as the offset or target pose (or position); the terms "offset" or "target" will be used interchangeably.

Notice here that this portion of the model as both rotated and deformed with respect to the reference position, and thus each of the vertices 104 has moved with respect to the reference pose.

[0022] The model data defining these poses is stored in respective data files, which will likewise be referred to as the reference geometry file, and the offset geometry file. An animation sequence may have any number of frames, depending on the length of the sequence, and for each such frame there is provided a geometry file for the model defining the pose of the model at the frame time. In the context of the present invention, one of these frames is selected as the reference frame. The model associated with such reference frames will be the reference geometry model. One of the features of the invention is the ability of the user to trade off compression ratio and geometry accuracy. The amount of compression increases as the value of quantization increases (i.e. a high error tolerance produces a high compression ratio). The frames and associated models following the reference frames are the offset frame and offset models respectively.

[0023] Fig. 4 illustrates a general software architecture for a system to provide compression of animation models. Generally, a model database 400 stores the geometry files for the animation sequence. Each geometry file is associated with a particular frame in the sequence, and has a file name that uniquely identifies the file; additional identifying information is typically stored in metadata for the file. A surface geometry compressor 402 reads reference geometry and offset geometry files from the database 400, and generates compressed geometry data. The compressed geometry data may be understood to express the offset file as the differences between predictions of the positions of the control vertices of the offset pose and the actual positions of the control vertices in the reference pose. The quantized prediction differences (deltas) are then provided to the data

compressor 404 which performs a further level of byte reduction. The data compressor 404 can be any available general (that is, data independent) compression algorithm. In one embodiment the data compressor 404 is a Lempel-Ziv type compressor, such as gzip. The resulting compressed file is stored back to the model database. A complementary geometry decompressor 406 and data decompressor 408 provide the corresponding decompression functions, and thus can be used to decompress the files when needed to render the models into the final animation sequence.

[0024] Fig. 5 illustrates the software architecture of the system in more detail, and will be referred to in the following discussion of the compression aspects of the present invention. Generally, the geometry compressor 404 takes as inputs a reference geometry file 502, an offset geometry file 502, and the name of the compression file 520 into which it will store the resulting compressed model information. The geometry compressor 404 also takes as an input a quantization value, which represents the maximum tolerance for error for each control vertex.

[0025] The geometry compressor 404 first verifies 506 the relationship between the reference geometry file 502 and the offset geometry file 504, specifically that these files are for the same model. This is done checking that the files include the same model name, the same number of surfaces, and the same number of control vertices within each surface; this information is typically stored in the metadata for the geometry files.

[0026] Next, the geometry compressor 404 traverses each surface sequentially in the reference geometry file 502, and locates the corresponding surface in the offset geometry file 504. Each surface has a unique name, which is used to match the surfaces between the files. The following steps of the compression operation take place within the scope of this traversal, so that for each surface the geometry

compressor 404 produces a corresponding compressed representation that is included in the compressed model 522.

[0027] As noted above, each surface comprises a set of points or control vertices, forming a rectangular matrix P[i,k]; each control vertex has a unique ID. The geometry compressor 404 traverses 510 over the set of control vertices in the reference geometry file 502, and uses the control vertex ID to locate the corresponding offset vertex $O_{i,k}$ in the offset geometry file 504 (we will use $O_{i,k}$. when referring to the actual value of the offset vertex retrieved from the offset geometry file, and $P'_{i,k}$ when referring to the predicted offset vertex). The traversal pattern is preferably in a zig-zag pattern as illustrated in FIG. 6. This traversal process proceeds along the first two rows of vertices, as illustrated by points $P_{0,0}$ to $P_{0,k}$ for the first row, and points $P_{1,0}$ to $P_{1,k}$ in the second row. When the last vertex in the second row $P_{1,k}$ is reached, the next two rows P_2 and P_3 are traversed, and so forth. If there are an even number of rows, than the traversal ends with the last element of the last even row. If there are an odd number of rows, then the last row (shown as $P_{i,0}$ to $P_{i,k}$ in FIG. 6) is traversed directly across.

[0028] During this traversal step, the geometry compressor 404 first selects a set of seed vertices. The seed vertices are used as the basis for prediction for other vertices in the same or nearby rows of points. As such, these vertices are themselves not compressed. At least the first three points in the traversal are used as seed vertices, but the selection of other seed points may be made based on algorithmic or implementation efficiency. FIG. 7 illustrates the seed vertices for a typical surface, here the vertices comprise the first two columns of the surface, excepting the last row. In this approach, a set of seed vertices is selected for each pair of rows. Alternatively, a single common set of seed vertices may be selected for the entire surface, for example, the four vertices in the upper left corner of the surface. The use of seed vertices for each pair of rows reduces

overall compression slightly, but also reduces the amount of compression error that can occur when compression and decompression takes place on computer systems with different floating-point processors.

[0029] The selected seed vertices are quantized 518, and passed to data compressor 406, which compresses 524 them and stores them in the compressed model file 522. The quantization function divides each coordinate value of the vertex by an error tolerance, rounds to the nearest integer, and multiplies by the tolerance value again. In one embodiment, the quantization function quantizes the floating-point representation of the vertices by truncating the mantissa of the floating-point number such that the absolute value of (quantized number – original) is less than an error tolerance.

[0030] After selecting the seed vertices, the geometry compressor 404 continues with the traversal and selects the next vertex on the surface. For example, if the first two columns are selected as the seed vertices, then the third vertex in the first row, $P_{0,2}$ will be next selected. Reseeding every two rows also reduces the propagation of error when used on different architecture.

[0031] The geometry compressor 404 next identifies 512 a basis triangle for the selected vertex, where the basis triangle comprises other control vertex on the surface. A basis triangle on the offset model is also selected. The basis triangle will be used to determine the relative position of the selected vertex with respect to the surface, using the basis triangle as local coordinate frame of reference. The basis triangle may be selected in various different ways from the set of vertices on the surface consisting of seed points and points that have already been traversed by the algorithm. The basis triangle is selected so that each of its vertices are geometrically nearby the point being compressed.

[0032] One way to select the basis triangle is find the closest triangle to the selected vertex. Another way to select the basis triangle is to select the triangle

based on a predetermined pattern that determines the closest triangle based on which row of the surface the vertex is in. FIG. 8 illustrates this approach.

[0033] In FIG. 8 on the left side, each vertex belongs to one of five regions designated as A, B, C, D, and E. Each vertex in regions B, C, D, and E have a nearby basis triangle. The vertices within each region share in common a pattern to identify the nearby basis triangle as illustrated on the right.

[0034] Area A are the seed vertices. For these vertices no basis triangles are selected.

[0035] Areas B are the odd rows, excepting the last row E. For control vertices $P_{i,k}$ in odd rows, the basis triangle 800 is selected as the vertices:

[0036] Basis triangle for odd rows: $P_{i,k-1}$, $P_{i+1,k-1}$, $P_{i+1,k-2}$.

[0037] Areas C are even rows. For control vertices $P_{i,k}$ in even rows, the basis triangle 800 is selected as the vertices:

[0038] Basis triangle 800 for even rows: $P_{i-1,k-1}$, $P_{i,k-1}$, $P_{i,k-2}$.

[0039] Area D is a singular case, the left most vertex in the last row of the surface, if the last row is odd. For this control vertex the basis triangle 800 is selected as the vertices:

[0040] Basis triangle 800 for vertex D: $P_{i-1,k}$, $P_{i,k+1}$, $P_{i-2,k+1}$.

[0041] Finally, for the last row of the surface, if it is odd, area E, the basis triangle 800 is selected as the vertices:

[0042] Basis triangle 800 for last row, excepting vertex D: $P_{i-1,k-1}$, $P_{i-1,k}$, $P_{i-2,k}$.

[0043] The use of the foregoing basis triangles based on the row of the selected vertex location with the zig-zag traversal method by which the geometry compressor 404 traverses the surface. More particularly, the basis triangle pattern ensures that for any given vertex, all of the vertices of the basis triangle will either be seed vertices or previously traversed.

[0044] A basis triangle defines a local coordinate system that is used to predict 514 the position of the selected control vertex on the offset surface, that is, the offset vertex. The basis triangle <A, B, C> defines two direction vectors, t and s, where t=C-A, and s=B-A, and a normal vector r, which is perpendicular to t and s and has a magnitude equal to the average of $|t|$ and $|s|$. r can be defined as follows:

[0045] $$\vec{r} = \frac{\vec{t} \times \vec{s}}{\left|\vec{t} \times \vec{s}\right|}$$

[0046] FIG. 9 illustrates these relationships on the reference surface showing basis triangle <A, B, C>, and on the corresponding offset surface, where there is an offset basis triangle <A', B', C'>, with corresponding direction vectors t' and s', and normal vector r'. A given control vertex $P_{i,k}$ is shown as well for the reference surface, and the actual location of the corresponding offset vertex $O_{i,k}$.

[0047] The geometry compressor 404 predicts 514 the position of the offset vertex $P'_{i,k}$ from the vertices of the basis triangle on the reference surface. In one embodiment, this is done as follows. As a preliminary, the control vertices A, B, and C have coordinates in the world space of the model, which coordinates can be represented as a vector. First, the selected control vertex $P_{i,k}$ is located on the reference surface in terms of t, s, and r. Next, this coordinate location is multiplied by the offset basis triangle's coordinate frame, t', s', and r'. This yields the predicted location of the offset vertex $P'_{i,k}$.

[0048] Mathematically, this process can be described by the following:

$$\vec{P'}_{i,k} = \left(\vec{P}_{i,k} - \vec{A}\right) * \begin{bmatrix} \vec{s} \\ \vec{t} \\ \vec{r} \end{bmatrix}^{-1} \begin{bmatrix} \vec{s'} \\ \vec{t'} \\ \vec{r'} \end{bmatrix} + \vec{A'} \qquad \text{(Eq. 1)}$$

where $\begin{bmatrix} \vec{s} \\ \vec{t} \\ \vec{r} \end{bmatrix}$ denotes a 3x3 matrix representing the coordinate frame derived from the basis triangle <A, B, C>, and $\begin{bmatrix} \vec{s} \\ \vec{t} \\ \vec{r} \end{bmatrix}^{-1}$ is its inverse, where A, A', P, and P' are row vectors and all defined in world space for the model.

[0049] Given the predicted offset vertex $P'_{i,k}$, the geometry compressor 404 next determines 516 a prediction delta $\Delta_{i,k}$ by subtracting the prediction $P'_{i,k}$ from the actual offset vertex position $O_{i,k}$, which value was previously retrieved from the offset geometry file 504. The prediction delta $\Delta_{i,k}$ is then quantized 518, and then added back onto the predicted offset vertex $P'_{i,k}$. The combined value is stored in memory, and will be used as needed as a vertex of a basis triangle <A', B', C'> on the offset surface when the geometry compressor 404 reaches control vertices later on the surface. If any of the coordinate frames are degenerate (i.e., the determinant of either of the 3x3 matrices equals 0), then the following equation is used:

$$\vec{P'}_{i,k} = \left(\vec{P}_{i,k} - \vec{A}\right) + \vec{A'} \qquad \text{(Eq. 2)}$$

[0050] The above prediction process continues over all of the control vertices for the current surface. The resulting data will be a set of seed point coordinates and the quantized prediction deltas Δ. Seed point coordinates are passed directly to the data compressor 404 to compress and store in the compressed geometry file 522. Prediction deltas are passed to a data reordering process 520. The reordering process 520 reorders the coordinate data of prediction deltas so that each vector component is stored contiguously. Thus, given that the deltas have

coordinates $<x_1, y_1, z_1>$, $<x_2, y_2, z_2>$...$<x_n, y_n, z_n>$ first all of the x components are collected (in the same order as the vertices), then all of the y components, and finally all of the z components. The resulting array $<x_1, x_2,...x_n, y_1, y_2,...y_n, z_1, z_2...z_n>$ is then passed to the data compressor 406, which compresses the bytes and stores it to the compressed geometry file 522. The data reordering takes advantages of the entropy encoding of the data compressor by clustering the component values together.

[0051] This process is repeated for each surface in the reference geometry file 502. The compressed coordinate strings for the surfaces are stored in the compressed geometry file 522 in the same order as the surfaces in the reference model, and thus there is no need to store the name of each file with the deltas. In addition, the seed vertices for each surface are also stored. When the compression for a single offset model is completed, the process is again repeated for each subsequent offset pose, for as many as desired by the animator. The resulting set of data, comprising the reference geometry file 502 and sequence of compressed geometry files 522, represents a portion of the animation sequence.

[0052] The geometry decompressor 406 and data decompressor 408 are used to decompress the compressed geometry files 522. This can be done anytime the underlying models for the animation sequence are needed, for example, for a rendering pass. Generally, the approach iterates over each surface in the reference geometry file, retrieves and uncompresses the seed point coordinates and mangled delta coordinates from the compressed geometry file, reorders the array to obtain the prediction deltas, then uses seed vertices to predict the control vertices, adjusting the predicted location with the corresponding prediction deltas. In one embodiment, the process is detailed as shown in FIG 10.

[0053] A given compressed model 522 is retrieved for decompression, along with a reference geometry file 502. The geometry decompressor 406 verifies 1002 the model relationships, again based on metadata in the respective files. The geometry decompressor 406 traverses 1004 each surface (compressed coordinate string) sequentially in the compressed file 522, and creates 1006 a corresponding new surface and associates it with an output offset geometry file. The duplicate surface will be referred to as the offset surface; this is the surface that will be reconstructed from the compressed geometry. The use of the reference geometry file 502 to reconstruct any of the offset geometry files 504 in the animation cycle is beneficial since it substantially reduces network traffic between the data servers storing the model data and the compute servers performing the decompression. This is particularly true in a rendering environment that uses network caching to store the reference model locally on the rendering computer, instead of having to repeatedly download the reference model. It is further beneficial as it eliminates the need to decompress any intermediate models between the reference geometry file 502 and the selected compressed model 522. In other words, decompression of an Nth compressed model does not depend on the decompression of the N-1th model. In contrast, other approaches typically rely on sequential decompression of models. The geometry decompressor 406 next retrieves 1008 the seed coordinates from the compressed file 522 for the current surface and the compressed prediction deltas Δ. The geometry decompressor 406 calls the data decompressor 402 to decompress 1010 the predicted deltas Δ, and then reorders 1012 the data back to its original <x,y,z> tuple form. At this stage all of the prediction deltas Δ for the current surface are available in memory.

[0054] The geometry decompressor 406 next traverses 1014 each control vertex $P_{i,k}$ in the reference geometry file 502 in the manner described above with

respect to FIG. 6, by zig-zagging across the rows of the surface. The geometry decompressor 406 identifies the seed vertices of the reference surface, again in the manner described above, as for example in conjunction with respect to FIG. 7. The seed vertices from the compressed geometry file 522 are copied to the corresponding vertices in the new offset surface.

[0055] Using the selected control vertex $P_{i,k}$ from the reference geometry file 502, geometry decompressor 406 selects 1016 a basis triangle, as described above, for example in conjunction with FIG. 8. As before, the vertices of the basis triangle will be either seed vertices or control vertices that have already been traversed. The geometry decompressor 406 also determines the corresponding basis triangle on the offset surface.

[0056] Next, the geometry decompressor 406 predicts 1018 the position of the offset vertex $P'_{i,k}$ in the offset surface using the coordinate system defined by the selected basis triangle, as described above with respect to FIG. 9 and Equation 1. As before, the coordinates of the offset basis triangle will be either seed vertices or vertices that have been previously traversed.

[0057] The geometry decompressor 406 then adds 1020 the corresponding prediction delta $\Delta_{i,k}$ to the predicted position $P'_{i,k}$ to obtain the final offset surface vertex O'.

[0058] This process is repeated for each control vertex for the surface, and the surface is written out the offset geometry file. When all of the surfaces have been traversed, the model has been reconstructed.

[0059] In the embodiments of the invention thus described, compression is achieved by storing the error of the prediction, rather than the underlying coordinate data itself. Generally these errors are small, and small numbers can pack into fewer bits of data than large numbers. Thus, the better the prediction function performs, the smaller the errors are, and the compression becomes

better. Those of skill in the art then can readily devise alternative prediction functions that predict each offset vertex using the coordinate information of other vertices in the same surface as control vertex being predicted.

[0060] The order that control vertices are processed in the foregoing embodiments, although beneficial, is not essential to the operation of the invention. It is possible to traverse the control vertices in a different way than described. For instance, a hierarchical traversal may subdivide a lattice of points into quadrants or halves, and then subdivide again, and so on until every point has been traversed. The same prediction function can be used as long as the points in the offset triangle have either been stored or have already been traversed. The prediction function in this embodiment would take the four corners of the current quadrant and predict the intermediate center location in order to subdivide the quadrant.

[0061] The order or pattern of traversal may depend on the type of model. In the above embodiments for a NURBS model, the zig-zag traversal pattern of FIG. 6 is beneficial. For subdivision and polygon model, a triangle-based traversal pattern may be used instead. In this approach, polygons arranged into triangle strips or triangle fans can be traversed in the same order that they are defined, as long as the reference geometry and the offset geometry share the same number of points and connectivity of triangles.

[0062] The present invention has been described in particular detail with respect to one possible embodiment. Those of skill in the art will appreciate that the invention may be practiced in other embodiments. First, the particular naming of the components, capitalization of terms, the attributes, data structures, or any other programming or structural aspect is not mandatory or significant, and the mechanisms that implement the invention or its features may have different names, formats, or protocols. Further, the system may be implemented via a

combination of hardware and software, as described, or entirely in hardware elements. Also, the particular division of functionality between the various system components described herein is merely exemplary, and not mandatory; functions performed by a single system component may instead be performed by multiple components, and functions performed by multiple components may instead performed by a single component.

[0063] Some portions of above description present the features of the present invention in terms of algorithms and symbolic representations of operations on information. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. These operations, while described functionally or logically, are understood to be implemented by computer programs. Furthermore, it has also proven convenient at times, to refer to these arrangements of operations as modules or by functional names, without loss of generality.

[0064] Unless specifically stated otherwise as apparent from the above discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system memories or registers or other such information storage, transmission or display devices.

[0065] Certain aspects of the present invention include process steps and instructions described herein in the form of an algorithm. It should be noted that the process steps and instructions of the present invention could be embodied in software, firmware or hardware, and when embodied in software, could be

downloaded to reside on and be operated from different platforms used by real time network operating systems.

[0066] The present invention also relates to an apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored on a computer readable medium that can be accessed by the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, application specific integrated circuits (ASICs), or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus. Furthermore, the computers referred to in the specification may include a single processor or may be architectures employing multiple processor designs for increased computing capability.

[0067] The algorithms and operations presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may also be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these systems will be apparent to those of skill in the, along with equivalent variations. In addition, the present invention is not described with reference to any particular programming language. It is appreciated that a variety of programming languages may be used to implement the teachings of the present invention as described herein, and any references to specific languages are provided for disclosure of enablement and best mode of the present invention.

[0068] The present invention is well suited to a wide variety of computer network systems over numerous topologies. Within this field, the configuration and management of large networks comprise storage devices and computers that are communicatively coupled to dissimilar computers and storage devices over a network, such as the Internet.

[0069] Finally, it should be noted that the language used in the specification has been principally selected for readability and instructional purposes, and may not have been selected to delineate or circumscribe the inventive subject matter. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.